POLITECHNIKA ŁÓDZKA

Wydział Elektrotechniki, Elektroniki,

Informatyki i Automatyki

Praca dyplomowa magisterska

Analiza problemu komiwojażera przy pomocy algorytmów heurystycznych

Traveling salesman problem analysis using heuristic algorithms

Łukasz Tracz

Nr albumu 234011

Opiekun pracy:

Dr inż. Paweł Marciniak,

Dr inż. Cezary Maj

Łódź, 2021

Streszczenie

Abstract

Spis treści

1. Wstęp

Obecnie transport oraz logistyka to dwa pojęcia bez których świat który znamy dzisiaj w ogóle nie mógł by powstać. Wszystkie państwa na świecie posiadają zawsze jakiś zasób w swoich granicach którego inny kraj w tej chwili wymaga. Taki stan rzeczy pozwolił na rozwój obecnie szeroko pojętego handlu czyli realizacji wymiany dóbr na konkretną ilość danego środka płatniczego. Przedmioty transakcji, zazwyczaj wymagają dostarczenia ich do miejsca przeznaczenia w stosownym czasie i poniesienia jak najmniejszego kosztu związanego z ich transportem.

Wówczas pojawił się jeden z najsłynniejszych z problemów optymalizacji który znamy dzisiaj pod ogólną nazwą „Problemu komiwojażera”. Jest to wyzwanie optymalizacji z ograniczeniami” tzn. jak znaleźć najlepszy układ zbiorów zmiennych przy znanych regułach i parametrach punktacji. Pomimo że ten problem znany był matematykom od lat to zainteresowanie nim dopiero wzrosło od lat 30 XX wieku. [1]Jednym z pierwszych zainteresowanych tym problemem który ponownie przedstawił to wyzwanie światu był Hassler Whitney z Harvardu, czego dokonał podczas swojego wykładu na Princeton University w 1934r. Po jego wystąpieniu publiczność z USA zapamiętała wyzwanie jako„ 48 stanowy problem Hasslera Whitneya”, ponieważ jego wersja problemu zakładała najbardziej optymalnej trasy pozwalającej odwiedzić wszystkie stolice 48 stanów (Alaska i Hawaje wtedy jeszcze nie były stanami). Problem ten był jeszcze badany przez wielu zdolnych matematyków i doczekał się naprawę wiele publikacji naukowych.

Pomimo że ten problem możemy uznać już obecnie za bardzo stary, to nadaj przez informatykę jest zaliczany do problemów NP-trudnych (znalezienie jego rozwiązania i jego weryfikacja nie jest możliwa w czasie wielomianowym). Dzisiaj dysponujemy zaawansowaną techniką o której XX wieczni naukowcy mogli jedynie pomarzyć. Liczba tranzystorów w procesorach z roku na rok zaczyna się powiększać w postępie wykładniczym, co zaobserwował jeden z założycieli firmy Inter Gordon Moor. Sformułował on w 1965r. empiryczne prawo mówiące, że liczba tranzystorów w procesorach podwaja się średnio co 2 lata. To tępo rozwoju jest jak najbardziej imponujące, jednakże rozwiązywanie tego samego problemu z wykorzystaniem większej mocy obliczeniowej w sposób taki sam jaki rozwiązywano go w XX w. nie przynosi nam podobne korzyści jak użycie większego młotka do wbicia tej same liczby gwoździ. Wówczas wniosek nasuwa się sam do rozwiązania tego problemu ludzkość nie koniecznie potrzebuje szybszych i wydajniejszych komputerów, ale potrzebuje lepszych sposobów na rozwiązanie go, tj. bardziej wydajnych algorytmów, algorytmów inteligentnych opartych na heurystyce i dokonań dość nowej dziedziny informatyki znanej pod pojęciem „Sztucznej inteligencji".

Ideą, jaka przyświeca tej pracy, jest zdefiniowanie problemu komiwojażera, omówienie dlaczego zaliczany jest do problemów NP-trudnych, opisanie znanych konwencjonalnych jak i wykorzystujących heurystykę metod oraz przeprowadzenie z ich użyciem pomiarów złożoności pamięciowej i czasowej z wykorzystaniem języka Python.

2. Motywacja

Główny motywem wyboru owego tematu przez autora było uzyskanie możliwości poznania już istniejących sposobów rozwiązywania problemu komiwojażera w oparciu o różne dokonania informatyczne i matematyczne poczynając od wykorzystania podstawowej wiedzy z teorii grafów i kombinatoryki aż po wykorzystanie programowania dynamicznego, algorytmów heurystycznych oraz algorytmów ewolucyjnych.

Autora również cechuje zamiłowanie do tworzenia oprogramowania i chęć ciągłego rozwoju dlatego jako język programowania do realizacji swoich badań wybrał on język Python, który pozwala na programowanie funkcjonalne jak i obiektowe oraz posiada dostęp do szerokiego zakresu gotowych bibliotek ułatwiających implementację algorytmów będących opisanych w tym artykule.

Pomimo, że przy rozwiązywaniu problemów optymalizacyjnych powinien być wybrany język, który jest uważany za najszybszy (obecnie C/C++), to autor zdecydował się wykorzystać język Python, z uwagi na jego wciąż rosnącą popularność oraz tym że w roku 2021 zgodnie z raportem „The software quality company" [2] wyprzedził on język C i C++ jak również zostawił on uwielbianą przez szerzę fanów tworzących aplikacje Enterprise Javę w tyle.

3. Cele i założenia pracy

Celem pracy było zaznajomienie się z istniejącymi już pracami badawczymi opisującymi różne podejścia do rozwiązania omawianego problemu oraz porównanie ich wydajności oraz optymalności uzyskanych z ich pomocą rozwiązań. Do rozpoczęcia pomiarów należało przygotować skrypt pozwalających na generowanie danych lokalizacyjnych dla N miast położonych w przestrzeni euklidesowej, oraz danych statystycznych o wykorzystanych koordynatach.

Przygotowanie danych wejściowych stanowił pierwszy etap pracy, kolejny etap zakładał wykorzystanie algorytmu typu Brute Force, by dla każdego zestawy wygenerowanych danych wejściowych zdefiniować optymalne rozwiązanie oraz minimalny koszt jego uzyskania. Aby zminimalizować błąd niedokładności pomiaru postanowiono dla każdej ilości miast przewidzianych do problemu zostały utworzonych po 100 prób posiadających N miast rozmieszczonych w przestrzeni euklidesowej o rozmiarach 2000x2000. Postanowiono również że każda próba zawsze będzie posiadała jeden zawsze położony w centrum układu punkt startowo-końcowy. Uzyskane w ten sposób dane zostaną wyeksportowane do plików JSON i będą stanowić dane wejściowe dla wszystkich algorytmów. Dla każdego przejścia pliku wejściowego został wygenerowany plik wyjściowy w formacie JSON zawierający drogę jaką miałby przebyć komiwojażer oraz koszt jej przejścia. Informacje tak uzyskane z algorytmów brutalnej siły autor planuje wykorzystać potem przy porównywaniu optymalności rozwiązań uzyskanych z użyciem algorytmów heurystycznych i ewolucyjnych.

Trzecim etap zakłada przeszukanie zasobów internetowych w celu znalezienia odpowiednich bibliotek do pomiaru zużycia pamięci oraz obliczania problemu komiwojażera z użyciem wybranych algorytmów. Opracowana została również też struktura plików JSON i skrypty w Pythonie do generowania danych pomiarowych. Do badań zdecydowano, że zostaną użyte jedne z najpopularniejszych algorytmów opracowanych do rozwiązywania problemów optymalizacyjnych. Poniżej przedstawiono ich zestawienie z podziałem na zasadę ich działania.

1. Algorytmy brutalnej siły
   a. naiwny algorytm siłowy
   b. algorytm Bellmana Helda-Karpa (programowanie dynamiczne)
2. Algorytmy zachłanne (prosta heurystyka)
   a. Algorytm najbliższego sąsiada
3. Algorytmy heurystyczne
   a. Algorytm A*
   b. Algorytm symulowanego wyżarzania
   c. Algorytm lokalnego przeszukiwania
   d. Algorytm genetyczny

Czwarty etap pracy zakłada przeanalizowanie i opracowanie zgromadzonych danych. Na tym etapie prac powinno nastąpić opracowanie skryptu którego zadaniem jest zamianę kolekcji plików pomiarowych w formacie JSON na plik CSV oraz uśrednienie zgormadzonych informacji dla danej liczby miast i użycia konkretnego algorytmu. Autor pracy zakłada że dane te pozwolą na potwierdzenie przypisanych już do omawianych w ramach innych prac badawczych im złożoności obliczeniowych. W tym etapie ma również zostać przeprowadzona analiza optymalności rozwiązań uzyskiwanych z użyciem różnych metod heurystycznych.

Piąty i ostatni etap stanowi miejsce na refleksje jakie wiążą się z możliwościami połączenia wykorzystania algorytmu zachłannego (dającego w większości przypadków mało optymalne rozwiązanie w krótkim czasie) z wykorzystaniem algorytmu genetycznego oraz algorytmu 2-optymalnego. Celem była implementacja algorytmu memetycznego oraz porównanie wydajności i optymalności uzyskanych z jego użyciem rozwiązań

4. Definicja problemu

Problem komiwojażera (ang. Traveling salesman Problem) stanowi problem obliczeniowy oparty o poszukiwanie w grafie cyklu Hamiltona o najmniejszej wadze. Zadanie to posiada liczne zastosowania w życiu codziennym. Jednym z jego najlepszych przykładów jest wiercenie obwodów drukowanych tj. każda maszyna ma na każdej z płytek do wywiercenia stałą liczbę dziurek do przytwierdzenia w ich miejsce elementów scalonych. Jeśli ramię robota zawsze będzie pokonywać najkrótszą trasę do wywiercenia tych otworów to fabryka będzie mogła produkować większą liczbę płytek niż w przypadku gdyby wiertło przemieszczało się zgodnie z ścieżką ruchu o dłuższej długości.

Obecnie w informatyce nie jest znany efektywny (tj. znajdujący rozwiązanie w czasie co najwyżej wielomianowym) algorytm, który zawsze zapewnia znalezienie optymalnego rozwiązania. Zagadnienie to zostało zaklasyfikowane do klasy problemów NP.-trudnych. Natomiast przy założeniu że poszukujemy cyklu o długości mniejszej niż x zaliczany jest wtedy do klasy problemów NP-zupełnych. Przy założeniu że rozważamy symetryczny problem komiwojażera (droga z puntu A do B jest taka sama jak z B d o A) przy grafie posiadającym N wierzchołków, liczba wszystkich możliwych cykli Hamiltona wynosi (N-1)!/2. Praktycznie sprawdzenie wszystkich tych kombinacji jest możliwe tylko wtedy gdy graf składa się z niewielkiej liczby wierzchołków.

Poniżej została przedstawiona tabela obrazująca relację między złożonością użytego algorytmu a ilością wierzchołków w grafie, dla którego zamierzamy znaleźć optymalne rozwiązanie.

Tabela 1 Złożoność obliczenia algorytmu kontra ilość danych wejściowych

n	n^2	n^5	2^n	n!
10	100	10 000	1 024	3 628 800
20	400	3 200 000	1 048 576	2 432 902 008 176 640 000
50	2500	312 500 000	1.12 x 10^{15}	3 x 10^{64}
100	10 000	10^{10}	1.2 x 10^{30}	9.3 x 10^{157}

4.1 Algorytm brutalnej siły

4.2 Algorytm zachłanny

4.3 Algorytm heurystyczny

5. Naiwne podejście siłowe

5.1 Algorytm Helda-Karpa

6. Podejście zachłanne

6.1 Algorytm najbliższego sąsiada

7. Algorytmy heurystyczne

7.1 Algorytm A*

7.2 Algorytm przeszukiwań lokalnych

7.3 Algorytm symulowanego wyżarzania

7.4 Algorytm genetyczny

8. Algorytm memetyczny

8.1 Algorytm 2-optymalny

9. Plan i zakres badań

10. Prezentacja wyników

11. Podsumowanie

12. Bibliografia

[1] University of Waterloo 200 University Avenue West Waterloo, Ontario, N2L 3G1, [Online]. Available: http://www.math.uwaterloo.ca/tsp/data/usa/tours.html.

[Data uzyskania dostępu: 18 11 2021].

[2] The software quality company, [Online]. Available: https://www.tiobe.com/tiobe-index/. [Data uzyskania dostępu: 19 11 2021].